3-D Hand Pose Estimation from Kinect's Point Cloud Using Appearance Matching

Pasquale Coscia, Francesco A.N. Palmieri,
Francesco Castaldo and Alberto Cavallo

Seconda Universitá di Napoli (SUN), Dipartimento di Ingegneria Industriale e dell'Informazione, via Roma 29, 81030 Aversa (CE) - Italy,
pasq.coscia@gmail.com, {francesco.palmieri, francesco.castaldo, alberto.cavallo}@unina2.it

Abstract. We present a novel appearance-based approach for pose estimation of a human hand using the point clouds provided by the low-cost Microsoft Kinect sensor. Both the free-hand case, in which the hand is isolated from the surrounding environment, and the hand-object case, in which the different types of interactions are classified, have been considered. The pose estimation is obtained by applying a modified version of the Iterative Closest Point (ICP) algorithm to the synthetic models. The proposed framework uses a "pure" point cloud as provided by the Kinect sensor without any other information such as RGB values or normal vector components.

Keywords: Hand Pose, Kinect, Point Cloud, RANSAC, DBSCAN, ICP

1 Introduction

Hand pose estimation, by means of one or more cameras, is a desired feature of many information systems in several applications, both in navigation and manipulation. For example in robotic applications, where robots are equipped with dexterous hands, it is important to provide feedback to the robot itself and/or to qualify how well it may be performing human-like grasps. In manipulators, efficient processing for analyzing hand motions may be an important part of a system specially when a device has to control complex machinery with many degrees of freedom (DoFs).

The possibility to use depth information with a relatively low-cost sensors has given new perspectives to the hand tracking [1, 2]. Among all, the Kinect sensor has become one of the most used depth sensor. The output of depth sensors is typically a point cloud, i.e. a data structure containing a set of multi-dimensional points expressed within a given coordinate system. In this work only the *xyz*-coordinates of the point clouds have been used. The point clouds have been stored at approximately 30 Hz and then processed.

The paper is organized as follows: Section 2 presents in detail each step of the proposed framework and their relative outputs. Section 3 shows experimental results for a number of real test sequences. The last section discusses the results and provides suggestions for improvements.

2 Our Approach

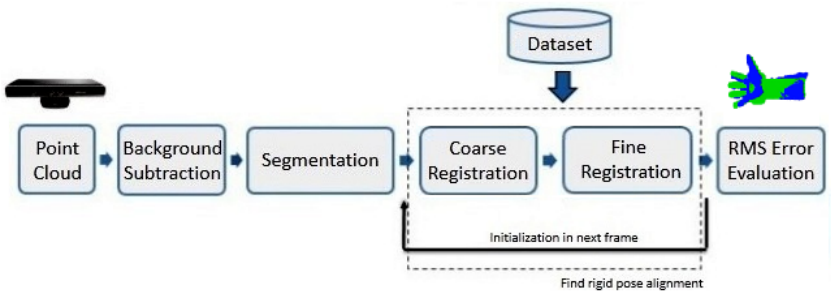


Fig. 1. Overview of the proposed framework for 3-D hand pose estimation.

Fig. 1 shows the block diagram of the proposed framework. The first step is background subtraction followed by clustering and matching steps in order to estimate the hand pose. Since the output of the matching step is fed back for the next frame, there is no real tracking. The starting point of the search, at each frame, is based on the results of the previous one. The pose is represented as a frame-dependent rotation matrix $\mathbf{R}$[k] and a translation vector $\mathbf{t}$[k] that update the position of the models' point clouds $\mathbf{M}_i$ in the space: $\mathbf{M}_i[k] = \mathbf{R}[k] \cdot \mathbf{M}_i[k-1] + \mathbf{t}[k]$.

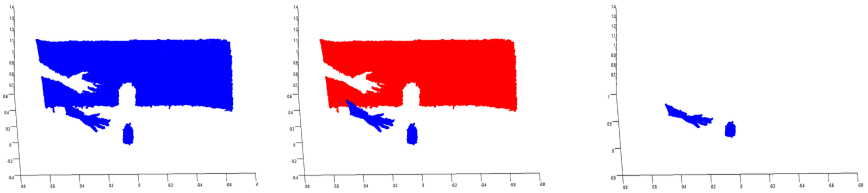
The point clouds have been captured using the Simulink Support for Kinect [3] and all the algorithms have been written in MATLAB. Each point cloud, before the processing, contains approximately 200k 3-D points. Our case study consists of a scene in which the hand is located in front of the sensor and has a plane background. For grasping tasks, we have considered only one object at a time, situated on a table. The hand and the objects are placed at a fixed distance. Therefore there is no need to change the scale of the models.

2.1 Background Subtraction

The first step in processing the data consists in removing the background, that in our case was roughly a plane (a wall). To perform the subtraction, we used the RANSAC algorithm. RANSAC stands for "Random Sample Consensus" and it is an iterative method for a robust parameters estimation of a mathematical model from a set of data points containing outliers. The algorithm is described in detail in [4]. In our case the model is represented by a plane, while outliers are the hand and the objects. In this work, we used the function written by Kovesi [5], that uses the RANSAC algorithm to robustly fit a plane to a set of 3-D data points.

Fig. 2 shows the result of the RANSAC algorithm during a grasping task. The background is completely removed without affecting the remaining point cloud represented by the hand and the object.

Background subtraction eliminates much of the data points in the cloud, hence it can be managed more easily and quickly for the following steps because the number of 3-D points typically drops by an order of magnitude.



(a) (b) (c)

Fig. 2. Results of applying RANSAC plane fitting algorithm: a) Original Point Cloud; b) Points (in red) that belong to the plane model determined by RANSAC; c) Point Cloud after background subtraction.

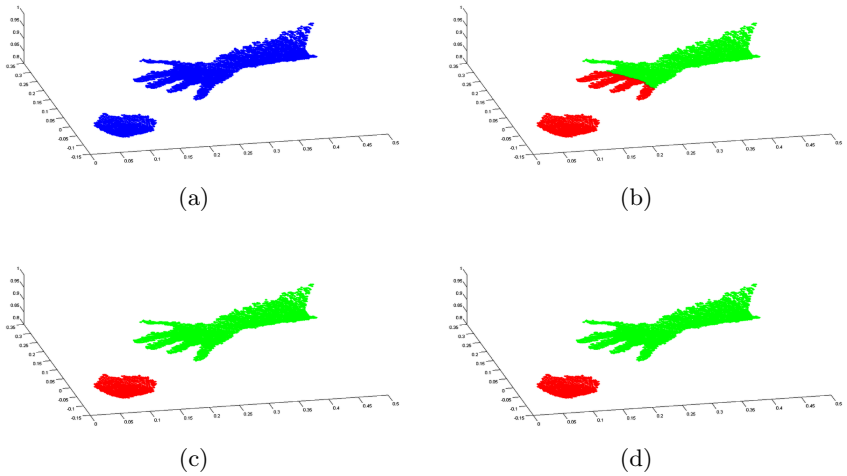
2.2 Segmentation

In order to associate each 3-D point to the corresponding cluster in the scene we have tested three different clustering algorithms: K-means, Euclidean Cluster Extraction and DBSCAN. K-means is a popular clustering algorithm that partitions data in a fixed a priori number of clusters. The Euclidean Cluster Extraction is a simple algorithm for clustering a 3-D point cloud using the Euclidean distance proposed by Rusu [6]. DBSCAN (Density based spatial clustering of application with noise) [7] is another technique based on density estimation for arbitrarily shaped clusters in spatial databases.

Fig. 3 shows the results of the segmentation step on a point cloud during a grasping task. More generally, if two clusters are close enough, K-means is not able to divide the objects correctly, whereas the DBSCAN and the Euclidean Cluster Extraction algorithms perform much better. In the following steps we have used the results of the DBSCAN algorithm that in our experiments has appeared to show the best performance.

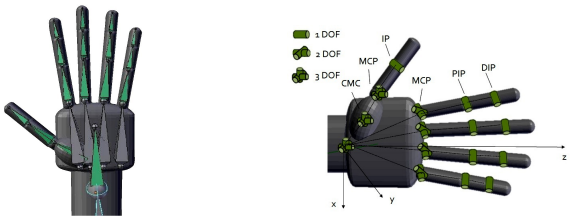
2.3 Synthetic hand models generation

The point clouds prototypes are obtained from the BlenSor software [8], which allows the creation of a point cloud from a 3-D model. A 3-D hand model with 26 DoFs has been built by assembling geometric primitives such as cylinders, spheres and ellipsoids. The shape parameters of each object are set by taking measurements from a real hand. In order to remain within feasible movements, hand and finger motion uses *static* constraints that typically limit the motion of each finger [9], see Fig. 4. In Fig. 5 some of the 3-D models are shown with their associated point clouds that have been used in building the prototypes for our dataset. In particular, we have considered three typical gestures of the hand (palm, fist and gun) and two "interaction" models.



(a) (b) (c) (d)

Fig. 3. a) Original Point Cloud; Output of the segmentation step by: b) K-means; c) Euclidean Cluster Extraction; d) DBSCAN.



(a) (b)

Fig. 4. a) 3-D hand model and corresponding skeleton that controls its deformations (constrained bones are shown in green); b) DOFs of the hand joints.

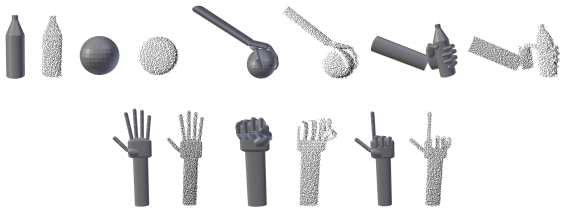


Fig. 5. The 3-D models and their corresponding point clouds generated for our dataset.

2.4 Matching

In order to perform the matching between the models and the data acquired by the Kinect sensor, the ICP algorithm [10] has been used. The algorithm at each iteration step, selects the nearest points of a cloud (the model, in our case) with respect to the other one (the input data) and computes the transformation, represented by a rotation matrix $\mathbf{R}$ and a translation vector $\mathbf{t}$, that minimize the following equation:

$$E(\mathbf{R}, \mathbf{t}) = \sum_{i=1}^{N_p} \sum_{j=1}^{N_m} w_{i,j} \|\mathbf{p}_i - (\mathbf{R}\mathbf{m}_j + \mathbf{t})\|^2, \tag{1}$$

where $\mathbf{M} = \{m_1, \ldots, m_{N_m}\}$ is the model's point cloud, $\mathbf{P} = \{p_1, \ldots, p_{N_p}\}$ is the input point cloud after the segmentation step and $w_{i,j} \in \{0, 1\}$ represents the point correspondences. A closed form solution for this rigid body transformation can be calculated using a method based on the singular value decomposition (SVD) [11].

Due to many problems, such as local minima, noise and partial overlap, several methods have been proposed to increase the robustness of the ICP algorithm [12] [13]. Our approach consists in adding a normally distributed noise and a random rotation matrix to perturb the position of the model's point clouds in order to obtain a good alignment at the first frame (a similar approach was proposed in [14]). More specifically: $\mathbf{m}'_j = \mathbf{R}_{rand}\mathbf{m}_j + \mathbf{s}, \quad j = 1, \ldots, N_m$, where $\mathbf{s}$ has zero mean and diagonal covariance matrix $\sigma^2 I$ and $\mathbf{R}_{rand}$ is a random rotation matrix. The parameter σ has been set to 0.1 for the experiments and its value has been determined experimentally. We have used the algorithm proposed by Arvo [15] to generate a random rotation matrix. The "stochastic" version of the ICP algorithm operates as follows: the model $\mathbf{M}$ is perturbed as discussed before, and then the standard version of the ICP algorithm on such perturbed model with respect to the input data $\mathbf{P}$ is applied. This operation is repeated a fixed number of times. Among all perturbed models, the one closest to the data captured by the sensor is chosen. Finally, the standard ICP algorithm is repeated on such model. In Fig. 6 the first four perturbed models obtained by applying the stochastic ICP algorithm are shown.

3 Experimental results

In order to evaluate the accuracy of the system, we have used the RMS error obtained by considering the Euclidean distance between each point of the selected model and the input data after the clustering step. In the next two sections, the experimental results of our approach are presented. In particular, we have tested the framework in two cases: performing static gestures and grasping an object located on a table. Fig. 7 shows the qualitative results in both cases and the corresponding RMS errors.

Static gestures: The hand is positioned approximately at a distance of 1.8 m from the Kinect sensor. In this case, the system is able to recognize the three

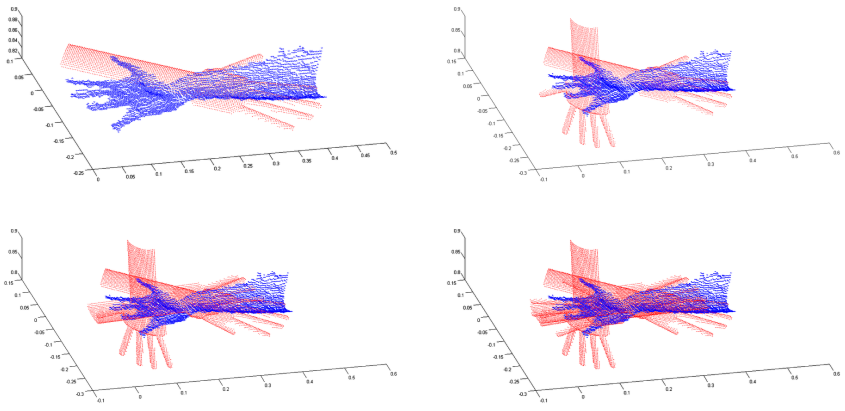


Fig. 6. First four perturbed models obtained by applying the "stochastic" version of the ICP algorithm. Among all perturbed models, the closest one to the data captured by the sensor is chosen.

models in the dataset. The RMS error is less than 10 *mm* and it remains constant during all the simulation since the gestures are distinguishable and correspond exactly to the synthetic models. When the hand moves by rotating, the error fluctuates more even if the correct pose is still recognized. In the last part of simulation, the error is greater and the wrong model is selected mainly because of a high rotation of the hand that no longer matches the models in the dataset. The visible part, which is captured by the Kinect, is effectively only the profile of the hand and a greater rotation of the model, even if it is correct, results in an increase of the RMS error.

Grasping objects Finally, we have tested our system when the hand interacts with an object. In particular, we have considered the grasping of objects easily obtainable in 3-D such as a ball and a bottle. For this experiments, we have replaced the three gestures (palm, gun, fist) with three intermediate poses representing the hand that comes close to the objects. The approach works quite well in the initial phase, recognizing the correct pose of the hand and the objects. As soon as the clustering algorithm returns a single cluster in the scene, only the interaction models are considered. As expected, the approach shows obvious problems in recognizing the correct model, when the clusters begin to join, because they are not completely merged. For the grasping ball task, the RMS error remains less than 10 *mm*, as in the above experiments. However, for some frames, the correct model is chosen but the estimated pose is wrong, primarily because of a lack of a real tracking. For the grasping bottle task some RMS error peaks are visible due to the presence of the entire arm that increases the surface area that needs to be considered for matching.

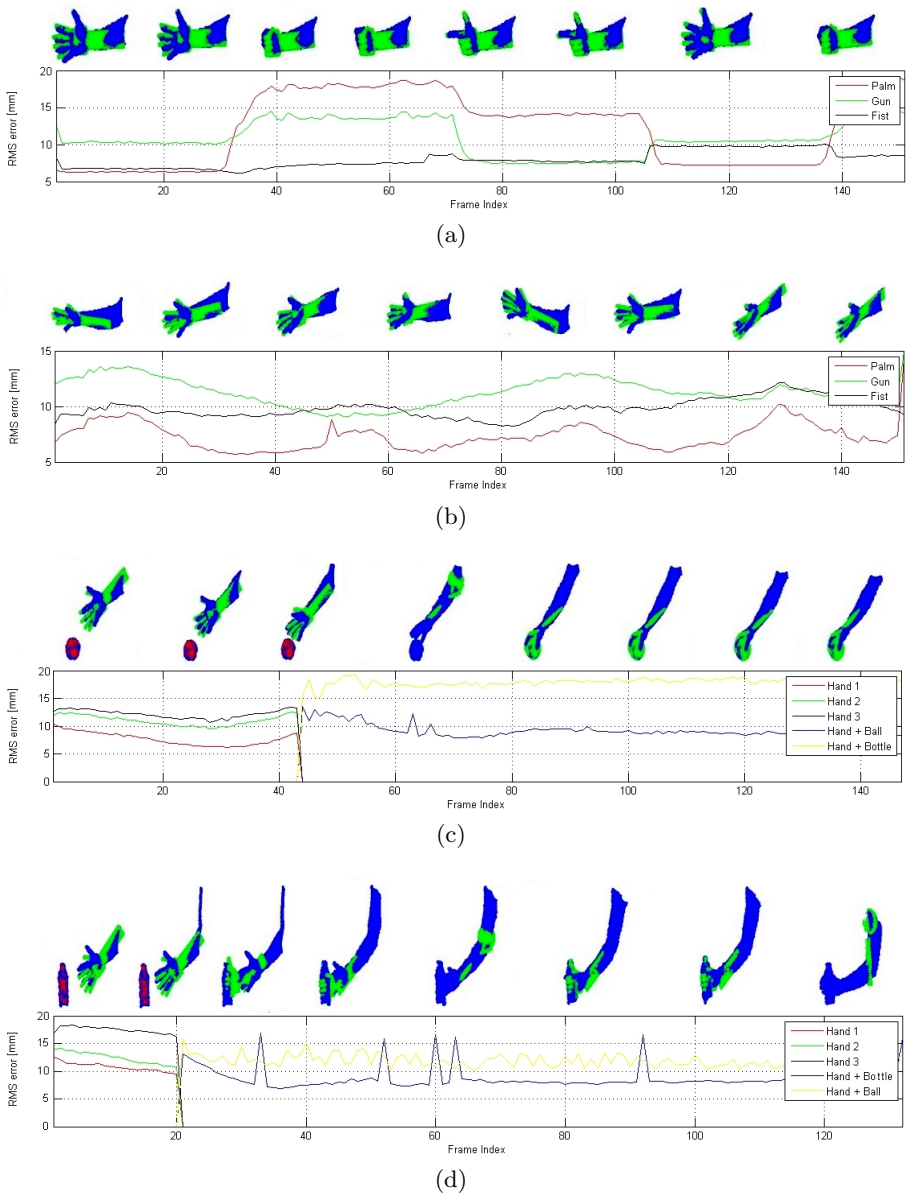


Fig. 7. Sample image sequences of the matching step and the corresponding RMS error: a) Hand performs static gestures; b) Hand performs rotations showing the palm; c) Ball grasping task; b) Bottle grasping task.

4 Conclusions and future work

In this work, we have presented a hand pose estimation system which is able to recognize a limited number of hand poses and interaction scenarios. The ex-

perimental results are very promising and demonstrate the effectiveness of the proposed approach. The system is able to provide a reasonable 3-D hand pose estimation using predefined models, as confirmed by the RMS error, also considering more complex scenarios in which the hand interacts with different objects. In our future work, possible improvements will be investigated. First of all, the system performance could be improved using parallelized code on GPU or by means of the use of the Point Cloud Library (PCL) in order to obtain a real-time system. Moreover, a larger dataset of typical gestures will be designed. Furthermore, an increase in accuracy could be achieved by implementing a tracking module to support the pose estimation.

References

1. Oikonomidis, I., Kyriazis, N., Argyros, A.: Efficient model-based 3D tracking of hand articulations using Kinect. Proceedings. 22nd British Machine Vision Conference, 101.1 – 101.11 (2011)
2. Melax, S., Keselman, L., Orsten, S.: Dynamics Based 3D Skeletal Hand Tracking. Proceedings. Graphics Interface Conference, 63 – 70 (2013)
3. Chikamasa, T.: Simulink Support for Kinect, `http://uk.mathworks.com/matlabcentral/fileexchange/32318-simulink-support-for-kinect`
4. Fischler, M. A., Bolles, R.C.: Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. Commun. ACM 24, 381 – 395 (1981)
5. Kovesi, P. D.: MATLAB and Octave Functions for Computer Vision and Image Processing, `http://www.csse.uwa.edu.au/~pk/research/matlabfns`
6. Rusu, R. B.: Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments. PhD Thesis (2009)
7. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. AAAI Press (1996)
8. Gschwandtner, M., Kwitt, R., Uhl, A., Pree, W.: BlenSor: Blender Sensor Simulation Toolbox. Proceedings of the 7th International Conference on Advances in Visual Computing, 199 – 208 (2011)
9. Lin, J., Wu, Y., Huang, T.S.: Modeling the constraints of human hand motion. Proceedings. Workshop on Human Motion, 121 – 126 (2000)
10. Besl, P.J., McKay, N.D.: A Method for Registration of 3-D Shapes. IEEE Trans. Pattern Anal. Mach. Intell. 14(2), 239–256 (1992)
11. Arun, K. S., Huang, T. S., Blostein, S. D.: Least square fitting of two 3-d point sets. IEEE Transactions on Pattern Analysis and Machine Intelligence 9(5), 698 – 700 (1987)
12. Langis, C., Greenspan, M., Godin, G.: The parallel iterative closest point algorithm. Proceedings. Third International Conference on 3-D Digital Imaging and Modeling, 195 – 202 (2001)
13. Li, C., Xue, J., Du, S., Zheng, N.: A Fast Multi-Resolution Iterative Closest Point Algorithm. Chinese Conference on Pattern Recognition (CCPR), 1 – 5 (2010)
14. Penney, G.P., Edwards, P.J., King, A.P., Blackall, J.M., Batchelor, P.G., Hawkes, D.J.: A Stochastic Iterative Closest Point Algorithm (stochastICP). 18th IEEE International Conference on Image Processing (ICIP) 2208, 762 – 769 (2001)
15. Arvo, J.: Fast Random Rotation Matrices. Graphics Gems III, Academic Press, 117 – 120 (1992)